Bug Tracker Workflow in GForge

Contents

The following is a description of the standard practice for using the GForge Tracker system for caCORE 3.x and initial 4.x releases.

General

- **Trackers**
  - **Support** - use only if extended support is needed, write it like a bug report (can be escalated to a bug by QA)
  - **Bugs** - Location where all changes to artifacts are recorded
    - To avoid duplicates and confusion, only post something after you have run it by the QA people.
    - Developers, QA, and Project Manager should use the query tool to make a view that shows only the bugs relevant to you or your situation.
  - **Feature Requests** - Location where new features can be requested by anyone with permission to post
    - Not used as forum, only with refined thought out requests
    - Once approved, Feature Requests can be escalated (moved) to the "Bug Tracker" via the "Data Type" field.
- **Forum** - Used to discuss feature requests, future builds, etc.

Life Cycle of a bug

- QA submits bug
  - QA must submit a bug for every change in the software (almost a task tracker). Thus, when a code change is finished, QA knows to test it.
  - Only submit a bug if you have run it by someone in the QA team
- Developer chooses highest priority bug and fixes it
- Developer then does the following:
  - Checks in code
  - Enters how they fixed the bug into the "Comments" field (if necessary)
  - Marks the bug's status "For Review"
  - Changes "Assign To:" to the QA that posted the bug
- A new build is made
- QA Tests the fixes
  - If fixed, change "State" field to "Closed"
  - (optional) change the "Assigned To" field to "Nobody"
  - (optional) change the "Status" to "Fixed"
  - If not fixed:
    - Change "Status" to "Open"
    - Add a comment to the bug that provides more detail on how it is still a problem
    - Change "Assigned To:" field to the developer

Build Query Tool

Since it is likely that we will have hundreds (maybe thousands) of bugs in the system, we need a way to view only the bugs we are interested in. The Build Query tool is the way to eliminate from your view any bug reports that you don't want to see. For example, a developer is not necessarily interested in bugs involving other developers or bugs from previous versions. To filter out only the bugs relevant to them for the current release, they create a custom query. To create their query, they would do the following

- Select the Bug Tracker
- Click on "Build Query"
- Select his/her name in the "Assign To:" field
- Select the current build (3.1 for example) for the custom field "Assigned Release"
- Select "Open" for the "State" field (you wouldn't want to see closed bugs because they are resolved).
- Select the "Save" button at the top

The query you made will be saved and can be used/modified at anytime. The query tool is particularly useful for a status meeting. You can

prepare a print out of only the current bugs in a certain modules (Product, category, etc.).

Bug Tracker

- **Phase** - Indicates the phase in which the bug was found
    - Integration Test
    - Integration Test - QA
    - System Test
    - System Test - QA
    - Internal User Acceptance
    - External User Acceptance
    - Release Candidate
    - QA Testing
    - User Acceptance Test
    - Production
- **Status** - this field, in combination with the "State" field, tell you exactly where the bug is in the life cycle of a bug.
    - None (automatically provided) - means that the bug is new and has not been addressed by the developer
    - Open - work in progress or was fixed then re-opened
    - For Review - developer marks this when:
        - Bug has been fixed by developer
        - Code is available to QA for verification (checked in)
        - Explanation/method for fix has been written in "Comments" field (if necessary)
        - Assigned To" has been changed to QA member that posted the bug
    - Cannot Reproduce - developer cannot reproduce the error
        - Developer changes the "Assign To:" to the QA that submitted the bug
        - QA needs to try to reproduce the bug. If they can't, then change the "State" to "Closed". If they can, mark "Status" to "Open" and add a comment or bring developer over to show them the bug.
    - Deferred - Developer marks this when they believe fixing this bug should wait till a future release.
        - Developer then marks why in the "How Fixed" text area. (optional)
        - Developer then changes the "Assigned To" field to the Project Manager or QA
        - If QA/Project Manager decides to not fix the bug in this release, then change the "Assigned Release" to "Future Release" and leave the "State" = "Open".
    - Invalid - bug is not a bug. That function works as it was intended.
        - Developer should send back to QA with comments explaining why invalid
        - If QA agrees, close the bug
    - Duplicate - bug exists somewhere else
        - Add comment giving bug # that it duplicates (ideally)
        - QA change the "State" to "Closed"
        - QA change "Assign To:" to "Nobody"
    - Release Note Candidate - indicates the issue needs to be documented as a known issue for that release
        - Once marked as a Release Note Candidate, Assign To the person that will change the documentation. DO NOT CLOSE THE BUG YET!
        - QA may need to re-enter a duplicate of the bug as a "Future Release" bug. That way it is recorded as a known issue for the current release, but a bug to be fixed for a future release also.
        - Once documents have been changed, close the bug
    - Awaiting Response - The bug is on hold till something happens to allow it to be worked on.
    - Fixed - (optional) Bug has been resolved or completed. This serves the same purpose as changing the "State" to "Closed" but is there for clarity. Thus it is optional.
        - Developers should not ever mark this (use "For Review" instead)
        - QA marks this only after bug has been confirmed fixed (Reviewed)
        - QA should change the "State" to "Closed"
- **Severity** - Indicates the effect of the bug on the system. NOTE: Though a bug may be major, like a crash, does not mean it will always be high priority and visa versa.
    - Critical - Causes a crash or data loss with no work around
    - Major - a bug that effects important functionality, No Work Around - No data loss
    - Normal - Work around exists
    - Minor - small effect on the software
- **State** (automatically provided)
    - Open - this bug has not been resolved or seen to its completion (Open by default)
    - Closed - ONLY QA or Project Manager can change this to closed! It means the bug has been resolved completely and doesn't need to be seen anymore.
- **Assigned Release** - The target release version for the bug to be fixed (i.e. 3.1)
- **Component** - This field separates out the program into traceable parts. Each project will need to add their own elements to this. Some projects may want to add another custom field that sub-categorizes the program. The purpose of this field is to allow us to view (using the "Build Query" tool) all the bugs within only a single portion of the software or track how many bugs were generated for one tool as compared to another.
- **Assign To** (automatically provided) - Person currently responsible for processing the bug
    - As long as the bug is still open ("State" = Open), the person marked here is responsible for processing it.
- **Priority** (automatically provided) - Assigned by QA initially, decided upon in group meeting with Project Manager, 1 - lowest, 5 - highest.
- **Summary** (automatically provided) - one sentence description
- **Detailed Description** (automatically provided) - Steps to reproduce, suggestions, and references not included in drop-down
- **Attach File** (automatically provided) - screen shots, etc.

- **File Description** (automatically provided) - description of attached file
- **Version Found** - Text field that gives the exact build revision label in which the bug was originally found (i.e. 3.1.2.9).
- **Previous Build Tested** - Since bugs are often re-opened, this text field provides the exact revision label that was tested last (not always the original revision label).
- **Comments** (automatically provided)
    - If necessary, Developers record how they fixed the bug (this helps QA verify the fix)
    - Enter comments on modifications made to the bug report. For example, "Changed priority to 5"
- **Type**
    - Code - error found in code using white box testing or code review
    - Cosmetic - requires only a minor change to the UI
    - Performance - software responds too slowly
    - Documentation - Error in some documentation (not code)
    - Usability - a behavior in the software that makes it hard to use
    - Incorrect Functionality - most common type of bug, use when an already specified function is not working as intended
    - Feature Request - an element to the program that has not currently in the program and has not previously been documented as a functional requirement.
    - Data Corruption - problem with the database
- **Estimated Resolution Time** - When the Developer first views the bug, they mark the estimated amount of time required to fix the bug. (Text Field)